# Embedded Applications Journal

A PUBLICATION OF INTEL CORPORATION FIRST QUARTER, 1994

## What's Inside



# The Intel386™ EX Microprocessor Core

Pranav Mehta
Senior Applications Engineer
Intel Corporation
Article ID# 0701

Intel Corporation announced its new family of embedded Intel386™ microprocessors in early October at the Fall 1993 Embedded Systems Conference in San Jose, California (see "Bringing DOS to Embedded Control" in the Q4, 1993 Embedded Applications Journal). This is the first in a series of articles that will examine, in more detail, various features offered by the flagship product of this family, the Intel386 EX microprocessor. This article offers an overview of the high-performance, 32-bit CPU core of the Intel386 architecture.

## The Core

The Intel386 EX microprocessor uses an enhanced 32-bit CPU core based on that of the standard Intel386 SX microprocessor. Thus, it is 100% object code compatible with the 80286, 80186, and 8086 microprocessors. The following sections describe the base Intel386 SX architecture followed by a list of enhancements.

## Base Architecture

As shown in Figure 1, the core of the Intel386 SX processor consists of a central processing unit, a memory management unit, and a core bus interface unit. The CPU consists of the execution unit and the instruction unit. The execution unit contains eight 32-bit general-purpose registers that are used for both address calculation and data operations and a 64-bit barrel shifter used to speed up shift, rotate, multiply, and divide operations. The instruction unit prefetches the instruction opcodes and stores them in the decoded instruction queue for immediate use by the execution unit.

The memory management unit (MMU) consists of a segmentation unit and a pag-

*Continued on page 3*

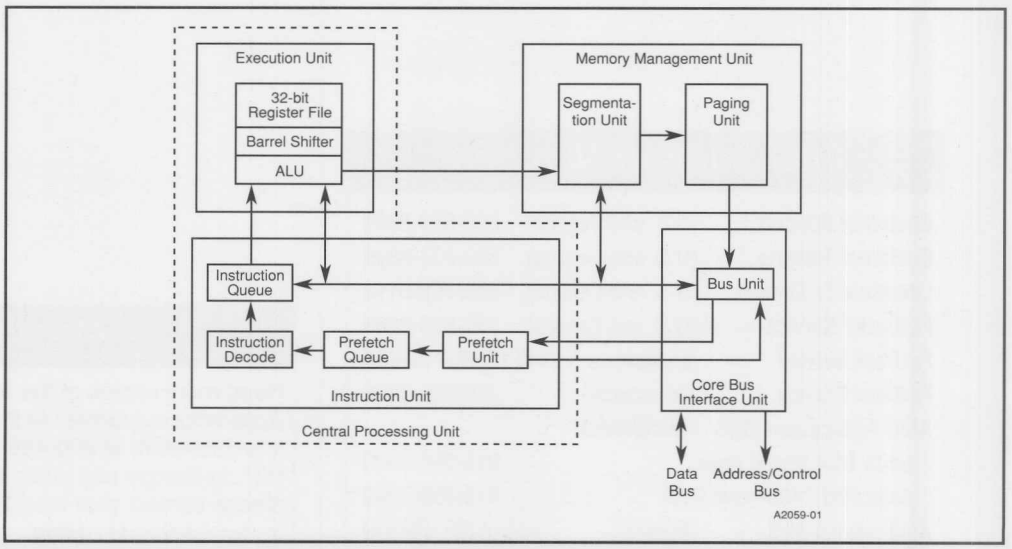


*Figure 1. Intel386™ SX Microprocessor Core*

# From the Managing Editor

In 1993 the Embedded Microcomputer Division (EMD) introduced several new products to the marketplace: the Embedded Intel386™ SX/CX and EX processors, the 3-volt 80C51 product family, and the 8XC196KD, MD, and NT, just to name a few. Some of these products extended our existing 8- and 16-bit portfolio, while others positioned EMD in the embedded 32-bit world by utilizing an industry standard architecture, the Intel386 CPU. Silicon products were not the only "firsts" for EMD this past year. The free Embedded Solutions Seminar (ESS I), held in the March–April time frame, was very successful. The Semiconductor Products Group (Embedded, Flash, Programmable Logic, and i960® technology) conducted seminars in 29 cities around the world, with over 4000 participants, demonstrating Intel's commitment to the embedded market. EMD also introduced a system modeling tool, Project Builder 196. This innovative tool models our embedded devices in real system applications, enabling customers to select a microcontroller based on system requirements. For the first time, the customer can test-drive a microcontroller in his application without having to know anything about that microcontroller.

As we move forward into 1994, we will continue to invest in our embedded microcontroller product line. You will see several new devices in 8-, 16-, and 32-bit bus widths, as well as new "design-in" tools to make it easier to learn, design, and produce reliable designs with our products. *Ap*BUILDER will be adding support for several 8- and 32-bit devices. Project Builder will continue to proliferate into the 8- and 16-bit products. Fuzzy logic will continue to grow, with the addition of NeuroFuzzy tools for all our devices.

In the April–May time frame, the Semiconductor Products Group will again sponsor the Embedded Solutions Seminar (ESS II). It will visit 17 cities in the U.S. and Canada, 12 cities in Europe, and the Asian Pacific Countries. With seminars in Embedded Intel386 processors, Flash memories, i960 components, Flex logic, and fuzzy logic, it promises to be an exciting time for all participants.

1994 marks the 15th year anniversary for the 8051 device. It's still going strong at the ripe old age of 15, shipping more than 75 units every minute and considered one of the "industry standard" 8-bit microcontrollers. We are committed to marching forward in embedded control and look forward to serving you for many years to come.

The Embedded Applications Journal is also committed to serving you! We read your suggestions and take them to heart. Stay tuned for articles on Flash memory and flash card designs, i960 processor basics, and fuzzy logic: friend or foe.

Steven M. McIntyre
Embedded Applications Manager
Embedded Microcomputer Division

## Intel Support Numbers

| | | |
|---|---|---|
| Customer Support | (U.S. and Canada) | 800-628-8686 |
| Customer Training | (U.S. and Canada) | 800-234-8806 |
| Literature Fulfillment | (U.S. and Canada) | 800-468-8118 |
| FaxBack* Service | (U.S. and Canada) | 800-628-2283 |
| FaxBack Service | (Europe) | +44 (0)793-496646 |
| FaxBack Service | (Worldwide) | 916-356-3105 |
| AMO Applications BBS | (Worldwide) | |
| up to 14.4 Kbaud lines | | 916-356-3600 |
| dedicated 2400-baud lines | | 916-356-7209 |
| Applications BBS | (Europe) | +44(0)793-496340 |

## More Copies

Need more copies of this issue of the Embedded Applications Journal? In the U.S. and Canada, call Intel Literature at 800-468-8118 and order #241294-007. In Europe and other international locations, please contact your local Intel sales office or distributor for additional copies.

ing unit. In the Intel386 architecture, addresses used by application/system programs to access memory have two parts: a base and an offset. This two-part address is referred to as virtual address. The segmentation unit translates a virtual address to a linear (or logical) address. A linear address is a 32-bit address. The paging unit then translates a linear address to a physical address. The size of the physical address depends on the number of address lines coming out of a processor. The Intel386 EX microprocessor has a 26-bit physical address, allowing it to address 64 Mbytes of physical memory.

The segmentation unit also provides four levels of protection for isolating and protecting applications and operating systems from each other. The hardware-enforced protection allows the design of systems with a high degree of integrity. Task management is another feature offered by the segmentation unit. Hardware-controlled task switching could be more efficient than software solutions if there is a close match between the task model of the Intel386 processor and that of the operating system.

The Intel386 microprocessor has two modes of operation: Real mode and Protected mode. In Real mode, the Intel386 processor operates as a very fast 8086, but with 32-bit extensions if desired. Protected mode offers all the sophisticated memory management, protection, and task switching functions. Within Protected mode, software can perform a task switch to enter into tasks designated as Virtual-8086 mode tasks. Each such task behaves with 8086 semantics, thus allowing 8086 software (an application or an entire operating system) to execute.

In addition to these powerful features, the Intel386 architecture provides features for debugging support via six debugging registers, DR0–3, DR6, and DR7.

## Core Enhancements

In addition to the standard features, the new embedded Intel386 processors incorporate the following enhanced features:

- The Intel386 EX microprocessor core is fully static, which means that even when the incoming clock signal is removed, the processor will retain its state.
- The Intel386 EX microprocessor core design is modular, allowing easier implementation of future proliferations of the embedded Intel386 microprocessor product family.
- The Intel386 EX microprocessor core is process-shifted from a 1 micron process to a 0.8 micron process, giving it more performance headroom. This will allow performance in the range of 3 – 7 MIPS and beyond over the next decade.
- Intel386 EX microprocessor is offered as a dual operating voltage part. This means that it operates at two supply voltage specifications, $V_{CC} = 5\ V \pm 10\%$ and $V_{CC} = 3\ V \pm 10\%$. In addition to low voltage operation (as low as 2.7 volts), the Intel386 EX microprocessor offers other sophisticated power management features that will be described in detail in a forthcoming article.
- Embedded applications are more likely to be exposed to various extremes of temperature than are standard desktop and portable PC applications. Components in an embedded design must be able to withstand this environment. The Intel386 EX microprocessor will be offered with an extended temperature range of –40° C to +85° C in the future.
- Another feature added to the Intel386 EX microprocessor core is the Intel System Management Mode (SMM). SMM is added as an additional operating mode in Intel Architecture along with Real, Protected, and Virtual-8086 modes. The SMM implementation of the Intel386 EX microprocessor is compliant with that of the SL-enhanced Intel486™ and Pentium® processors. Although originally introduced for power management features in portable computers, SMM can be used in the embedded environment for other purposes, such as debugging, having an alternate operating system, or virtualizing I/O devices. More technical information on this topic will be forthcoming.

## Summary

Intel's new family of embedded Intel386 microprocessors is powered by a fully static, 32-bit core. The higher performance and addressability of this Intel386 microprocessor core, coupled with low power and new enhanced features, enable integrated solutions that will make it the Embedded Processor of Choice for the 90's.

# Intel's Embedded Solutions Seminar

**sem-i-nar** \'sem-ə-när\ *n* [G. fr. L *seminarium* seminary]
**1.** an advanced or graduate course
**2.** a meeting for giving and discussing information
**3.** *see* Intel's Embedded Solutions Seminar

Joe Altnether
Senior Technical Marketing Engineer
Intel Corporation
Article ID# 0702

In the fast-paced world of embedded applications, you can't afford to fall behind. Staying current is an endless task of gathering information on the latest developments. Wading through the technical magazines is a time-consuming activity. Fortunately, there are other methods to obtain the information. One of these methods is the Intel Embedded Solutions Seminar series.

First presented last year, these seminars proved to be highly popular. The seminars provided a balanced mix of product information, application techniques, and a hands-on display suite that showcased the products and development tools necessary for successful embedded applications.

That was last year; much has changed in a year. As a result, the second Intel Embedded Solutions Seminar has been produced to provide the latest information on Intel's embedded products. This is not a rehash of last year's seminar, but a completely new seminar with new topics and information. To provide more information that's focused on your application needs, the seminar is structured in a university format. Choose only those topics that are pertinent to your areas of interest. Shown below is the agenda for the full-day seminar series:

| | |
|---|---|
| 8:30 – 8:50 a.m. | Welcome/Introduction |
| 9:00 – 10:20 a.m. | Applications (3 Parallel Sessions)<br>• PC Power in Embedded<br>• Data Control/Communications<br>• Event Control |
| 10:20 – 10:40 a.m. | Break |
| 10:40 – 12:00 noon | • i960® Microprocessor — The Ideal Processor for High-Performance Embedded Applications<br>• Embedded Intel386™ Microprocessors Bring New Capability to Embedded Applications<br>• Flash Memory Design Techniques |
| 12:00 – 1:30 p.m. | Lunch |
| 1:30 – 2:50 p.m. | • Designing with Microcontrollers<br>• FLEXlogic — FPGAs with a Difference<br>• Fuzzy Logic Lab |
| 2:50 – 3:10 p.m. | Break |
| 3:10 – 4:30 p.m. | • Flash Memory Design Techniques<br>• Fuzzy Logic Lab<br>• FLEXlogic — FPGAs with a Difference |

The display suite will be open during breaks, during lunch, and at the end of the day.

The day begins with a choice of one of three technical application sessions on embedded applications. Each session covers a specific application segment: system control, data control/communications, and event control. In these sessions, the problems and solution requirements unique to the segment are developed.

## PC Power in Embedded

In the segment, applications that rely on PC/DOS compatibility to achieve a standard user interface and a very short development cycle are discussed. Applications based on the Embedded Intel386™ microprocessor, Flash memory, and FPGAs are covered. Typical of this class of application are ROM-DOS-based handheld barcode scanners, Windows* point-of-sale terminals, and digital office peripherals using Microsoft* Windows at Work*.

## Data Control/Communication

The data control session describes high-performance data communications and data control applications incorporating the Intel i960® microprocessors, Embedded Intel386 microprocessors, Flash memory, and FPGAs. Typical applications include bridges, hubs, routers, SCSI, and RAID.

## Event Control

The event control session explores those applications that must respond to real-time events. It examines the physical solutions to event control problems, focusing on the hardware and peripheral functions of the Intel MCS® 51 and MCS 96 microcontrollers along with Flash and EPLDs. Control strategies and software solutions are also discussed. These include not only the conventional methods but also fuzzy logic and neural networks.

To support these technical discussions and provide more technical depth, additional sessions focus on the details of the products included in the application sessions. There are lectures on both the Intel microprocessors/controllers and the memory components that work with them to build embedded systems. The following is a list of the available sessions.

## Flash Memory Design Techniques

This session provides details on Intel's Flash products, including the latest 16-Mbit FlashFile™ architecture and the Flash cards and drives. Design techniques for the 2-Mbit and 4-Mbit boot-block Flash memories for highly integrated systems featuring safe firmware updates are presented. Also included are design techniques for the 8-Mbit and 16-Mbit FlashFile memories used for resident Flash arrays and resident Flash disks. An overview of Flash memory cards, Flash drives, and the required software is provided.

## Designing with Microcontrollers

The architectures of the MCS 51 and MCS 96 microcontrollers, the performance differences of the two architectures, and the peripheral functions at the system level are examined. New products incorporating low voltage and demultiplexed buses are presented, and demultiplexed versus multiplexed bus performance is analyzed. The available tools such as *Ap*BUILDER and Project Builder 196 are included in the discussion. Copies of the *Ap*BUILDER and Project Builder 196 software are provided.

## Fuzzy Logic Lab

This is a lab with instruction on the Inform MCU-96 Explorer fuzzy tool. The fuzzy theory basics that were presented in the morning session are expanded upon. This is implemented on the Intel embedded microcontrollers. Using the MCU-96, the linguistic variables are defined and the rules are written to control the movement of a container crane.

## i960® Processor — The Ideal Processor for High-Performance Embedded Applications

An overview of the i960® architecture and key features and members of this RISC processor family are presented. Application development process and various Intel and third party development tools are addressed. This includes the use of profiling compilers to enhance execution performance.

## FLEXlogic — FPGAs with a Difference

In this session, the FLEXlogic architecture and its implementation in the iFX780 and iFX740 devices are examined. A key focus is on the features of these devices as well as a brief introduction to features of new devices that can be expected later in 1994. The predictable nature of performance in the FLEXlogic family is shown. The design tools are reviewed with emphasis on Intel's PLDshell Plus design package. In this session, the attendees will develop a representative application using the FLEXlogic tool.

## Embedded Intel386™ Microprocessors Bring New Capability to Embedded Applications

Intel's new embedded Intel386™ processors are presented, including a detailed technical presentation on the integrated product, the Intel386 EX microprocessor. Various peripheral modules integrated on the Intel386 EX microprocessor are explained. PC/DOS compatibility issues with various operation modes of the Intel386 EX microprocessor are detailed. For real-time applications, Intel's iRMX® EMB operating system and its features are discussed. Migration from Intel386 SL and 80C186 microprocessor-based platforms to the Intel386 EX microprocessor are examined.

*Continued on page 6*

# Initializing the 80C186EC Interrupt Control Unit

Robin S. Manelis
Information Engineer
Intel Corporation
Article ID# 0703

The 80C186EC uses its interrupt control unit (ICU) to determine when and how to service its on-chip peripherals.

The main components of the interrupt control unit are two 8259A modules. One is hardwired to operate as the master; the other, as the slave. The slave 8259A is cascaded from the master 8259A's interrupt request line 7 (IR7). Each 8259A has seven registers used to initialize it and modify its operation during program execution. Programming the ICU can be confusing because you must access these seven registers through only two access ports.

To clarify the 80C186EC interrupt control unit initialization process, this article explains how to access the registers, how to place interrupt handler addresses into the interrupt vector table, and how to initialize the 8259As.

## Accessing the Registers

Each 8259A has a set of seven registers: four initialization command words (ICWs) and three operation command words (OCWs). However, each register set occupies only two locations in the I/O address map (0000H and 0002H for the master, 0004H and 0006H for the slave). To access a specific register, you must manipulate certain bits and write to the registers in a particular sequence. (See Table 1.) Use the ICWs to configure the modules during system initialization and the OCWs to modify the modules' operation during program execution.

## Placing Interrupt Handler Addresses in the Interrupt Vector Table

You need to write the addresses of your interrupt handlers into the interrupt vector table. The interrupt vector table is divided into four-byte sections called interrupt types. Each interrupt handler address requires

*Continued on page 7*

**Intel's Embedded Solutions Seminar**

*Continued from page 5*

## Display Suite

In addition to the lectures, a product and tools display suite is open during the seminar. Here examples of innovative end-user products, on-line engineering design tools, and third party development tools that support Intel's products and services are presented. Tools that simplify design and development of applications, including fuzzy logic, are on display. Microsoft's Windows at Work for embedded applications is shown. Selected third party representatives as well as Intel and distributor technical personnel will be available to answer your questions.

To make it easier to attend, we have expanded the seminar sites. More worldwide sites will be added later in the year. Listed below are the confirmed sites and dates for the seminars.

| | | |
|---|---|---|
| Tuesday, April 12 | Seattle, WA | San Diego, CA |
| Wednesday, April 13 | San Jose, CA | Orange County, CA |
| Thursday, April 14 | San Jose, CA | Sherman Oaks, CA |
| Tuesday, April 19 | Baltimore, MD | Orlando, FL |
| Wednesday, April 20 | Philadelphia, PA | Atlanta, GA |
| Thursday, April 21 | Boston, MA | Cleveland, OH |
| Tuesday, April 26 | Denver, CO | Minneapolis, MN |
| Wednesday, April 27 | Austin, TX | Chicago, IL |
| Thursday, April 28 | Dallas, TX | Toronto, Canada |
| Monday , May 2 | Stockholm, Sweden | |
| Tuesday , May 3 | Munich, Germany | Amsterdam, Netherlands |
| Wednesday , May 4 | Stuttgart, Germany | Brussels, Belgium |
| Thursday , May 5 | Düsseldorf, Germany | Paris, France |
| Friday , May 6 | Zurich, Switzerland | Milan, Italy |
| Monday , May 9 | Madrid, Spain | |
| Tuesday , May 10 | UK | UK |
| Monday , May 16 | Tel Aviv, Israel | |

If you attended last year's seminar, you will want to attend this one as well. If you missed last year's seminar, you can't afford to miss this one. Registration for the seminar is as simple as a phone call. To register, call 1-800-368-4110 or contact your local Intel distributor.

four bytes (one interrupt type) — two bytes for its IP and two bytes for its CS. Write the master interrupt handler addresses to eight consecutive interrupt types and the slave interrupt handler addresses to another eight consecutive interrupt types in the interrupt vector table. (See Table 2.)

The ICU uses the following formulas to determine a handler's address in the interrupt vector table.

INT Type x 4 = Location of handler's IP
INT Type x 4 + 2 = Location of handler's CS

*Continued on page 8*

*Table 1. Accessing the Initialization and Operation Command Words*

| Writing to | Accesses |
|---|---|
| 0000H with bit 4 set | ICW1m |
| 0002H immediately following a write to ICW1m | ICW2m |
| 0002H immediately following a write to ICW2m | ICW3m |
| 0002H immediately following a write to ICW3m | ICW4m |
| 0002H (**not** following a write to ICW1-3) | OCW1m |
| 0000H with bits 4 and 3 clear | OCW2m |
| 0000H with bit 4 clear and bit 3 set | OCW3m |

| Writing to | Accesses |
|---|---|
| 0004H with bit 4 set | ICW1s |
| 0006H immediately following a write to ICW1s | ICW2s |
| 0006H immediately following a write to ICW2s | ICW3s |
| 0006H immediately following a write to ICW3s | ICW4s |
| 0006H (**not** following a write to ICW1-3.) | OCW1s |
| 0004H with bits 4 and 3 clear | OCW2s |
| 0004H with bit 4 clear and bit 3 set | OCW3s |

*Note: A lower case 'm' indicates a master ICW or OCW; a lower case 's' indicates a slave ICW or OCW.*

*Table 2. Master and Slave Interrupt Handler Addresses*

| Write | To INT Type (Note 1) | At Location |
|---|---|---|
| Master's IR0 IP<br>Master's IR0 CS | Type X | X x 4 X x 4 + 2 |
| Master's IR1 IP<br>Master's IR1 CS | Type X + 1 | (X + 1) x 4<br>(X + 1) x 4 + 2 |
| Master's IR2 IP<br>Master's IR2 CS | Type X + 2 | (X + 2) x 4<br>(X + 2) x 4 + 2 |
| Master's IR3 IP<br>Master's IR3 CS | Type X + 3 | (X + 3) x 4<br>(X + 3) x 4 + 2 |
| Master's IR4 IP<br>Master's IR4 CS | Type X + 4 | (X + 4) x 4<br>(X + 4) x 4 + 2 |
| Master's IR5 IP<br>Master's IR5 CS | Type X + 5 | (X + 5) x 4<br>(X + 5) x 4 + 2 |
| Master's IR6 IP<br>Master's IR6 CS | Type X + 6 | (X + 6) x 4<br>(X + 6) x 4 + 2 |
| (Note 2) | | |

| Write | To INT Type (Note 1) | At Location |
|---|---|---|
| Slave's IR0 IP<br>Slave's IR0 CS | Type Y | Y x 4 Y x 4 + 2 |
| Slave's IR1 IP<br>Slave's IR1 CS | Type Y + 1 | (Y + 1) x 4<br>(Y + 1) x 4 + 2 |
| Slave's IR2 IP<br>Slave's IR2 CS | Type Y + 2 | (Y + 2) x 4<br>(Y + 2) x 4 + 2 |
| Slave's IR3 IP<br>Slave's IR3 CS | Type Y + 3 | (Y + 3) x 4<br>(Y + 3) x 4 + 2 |
| Slave's IR4 IP<br>Slave's IR4 CS | Type Y + 4 | (Y + 4) x 4<br>(Y + 4) x 4 + 2 |
| Slave's IR5 IP<br>Slave's IR5 CS | Type Y + 5 | (Y + 5) x 4<br>(Y + 5) x 4 + 2 |
| Slave's IR6 IP<br>Slave's IR6 CS | Type Y + 6 | (Y + 6) x 4<br>(Y + 6) x 4 + 2 |
| Slave's IR7 IP<br>Slave's IR7 CS | Type Y + 7 | (Y + 7) x 4<br>(Y + 7) x 4 + 2 |

*Notes: 1. X is the base interrupt type for the master 8259A; Y is the base interrupt type for the slave 8259A. You write these base types to the 8259As during initialization.*

*2. There is no master IR7 handler because the slave is cascaded from this line.*

*Continued from page 7*

## Initializing the 8259As

To initialize the 8259As, first globally disable all interrupts using the CLI command, then write to the initialization command words. (The ICWs are shown in Figure 1.) To initialize the master, write to its initialization command words in order (ICW1m, ICW2m, ICW3m, then ICW4m). To initialize the slave, write to its initialization command words in order (ICW1s, ICW2s, ICW3s, then ICW4s).

Use ICW1 to select edge-triggered or level-sensitive interrupt requests.

Use ICW2 to tell the ICU where you placed the interrupt handler addresses in the interrupt vector table. Write the base interrupt type to bits 7:0. For example, if you placed the master's IR0 handler address at type 70H, write 70H to ICW2m bits 7:0. The ICU adds the interrupt request line number to the base interrupt type, then presents the calculated interrupt type to the CPU. For example, with an interrupt request on the master's IR4 line and a master base interrupt type of 70H, the ICU sends interrupt type 74H to the CPU.

Use ICW3m to indicate which master IR lines have slave 8259As connected to them. Setting a bit indicates that a slave 8259A is connected to the corresponding IR line. Since the ICU has a slave 8259A cascaded from the master's IR7 line, you must set the ICW3m S7 bit.

Use ICW3s to indicate which master IR line a slave is connected to. Since the ICU's slave 8259A is cascaded from the master's IR7 line, you must write 7 to ICW3s bits 7:0.

Use ICW4 to select special fully nested mode or fully nested mode and to enable or disable the automatic EOI mode.

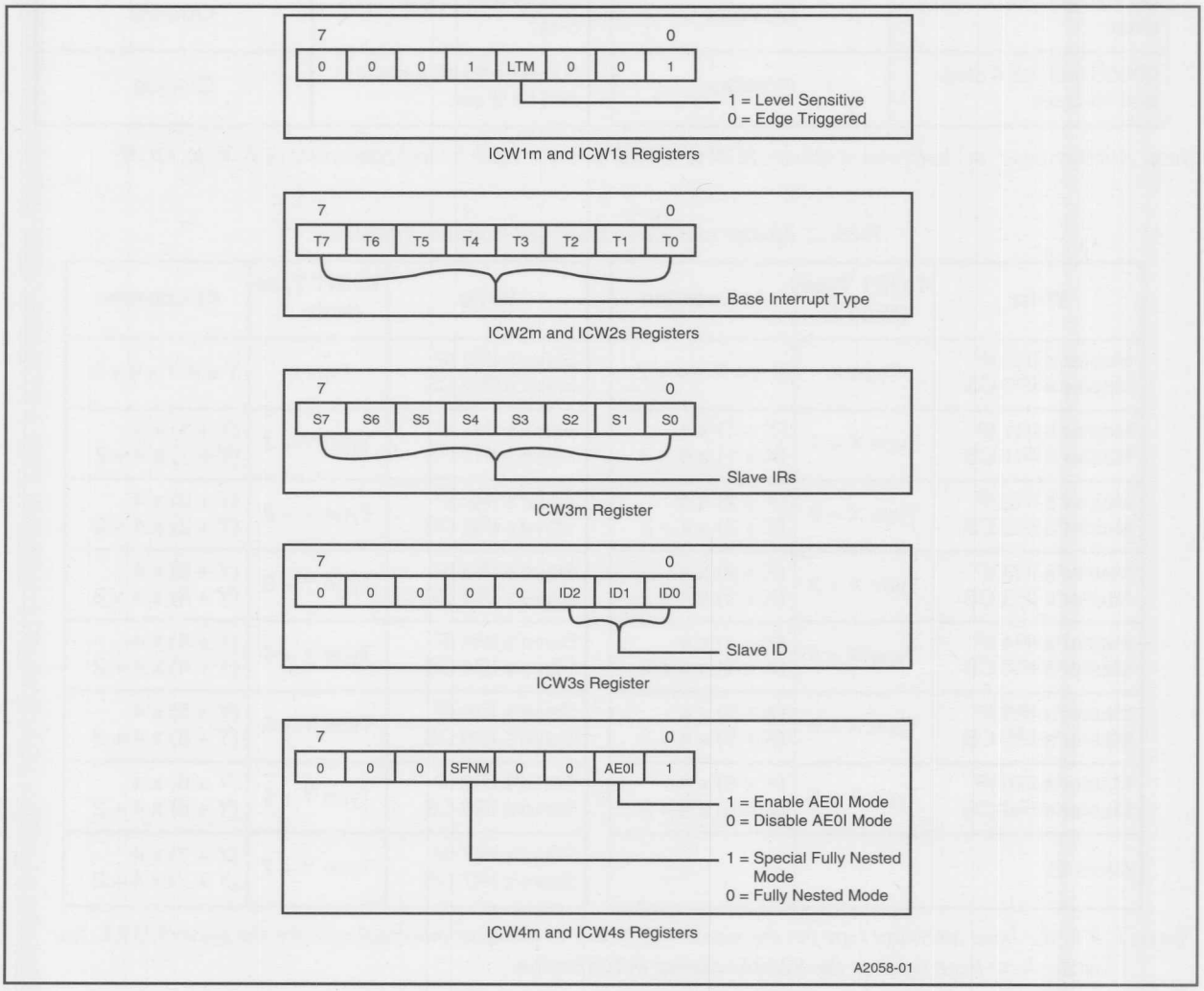


*Figure 1. Initialization Command Words*

# MCS® 96 Microcontrollers — The Perfect Match for Fuzzy Logic Applications

Mohammed Fennich
Applications Engineer
Intel Corporation
Article ID# 0704

In the domain of control systems, fuzzy logic proved very useful and very suitable for systems without mathematical models and systems with too much uncertainty. In a way, it came to the rescue when conventional methods started to reach dead ends.

However, fuzzy logic also has its down side. The major drawback of the emerging technology is the lack of a formal design methodology. The other disadvantage is that the resulting system is not analytical, and therefore any mathematical analysis on paper is impossible with current methods.

The lack of formalism at the application level, however, did not obstruct the continuous path of fuzzy logic. Even with this handicap, a large number of applications have been implemented using fuzzy controllers. The basic and essential requirements were found to be of a hardware and software nature.

When a conventional embedded microcontroller is used in conjunction with a software fuzzy logic algorithm, the rules that constitute the base of the algorithm are evaluated in a sequence, one after the other. Once all the rules are evaluated, their outputs are combined to provide a single value that will be defuzzified.

In contrast, if a dedicated fuzzy processor is used, the rules are evaluated in parallel. The parallel processing method suggests a fast processing cycle. However, in this case, data acquisition and data output still have to be done using conventional peripherals. The time gained in processing the rules in parallel can be lost in acquiring the data via external peripherals.

For most control systems, the best solution is to use a software fuzzy algorithm on a microcontroller with fast internal peripherals. In this case, the sequential rule-processing scheme is transparent to the user and the process seems to have been done in parallel.

The MCS® 96 microcontrollers are equipped with high-performance internal peripherals that make data acquisition, conditioning, and output fast and easy to handle. These peripherals, the wide range of addressing modes, and the powerful set of instructions that the family offers make these controllers very suitable for fuzzy logic applications.

The example below shows how some of the peripherals of the 80C196KD were used to acquire data via sensors in a closed-loop system with two inputs and one output. An overview of the hardware involved is given, and the steps taken in order to develop the software are also presented.

An experimental model representing a container crane was constructed. The system built consists of a pendulum of mass M and length L. The pendulum is hooked to a car that runs on rails. When the car moves from right to left or from left to right, the pendulum moves with it and swings.

The objective is to use fuzzy control to move the car from an initial position to a pre-assigned final position, and at the same time limit the oscillations of the pendulum. The block diagram of the system is shown in Figure 1.

An optical quadrature encoder is used as a position sensor. It has two outputs, channel A and channel B.

The sensor was mounted on the shaft of the motor. When the shaft turns, the sensor outputs the two square

*Continued on page 10*

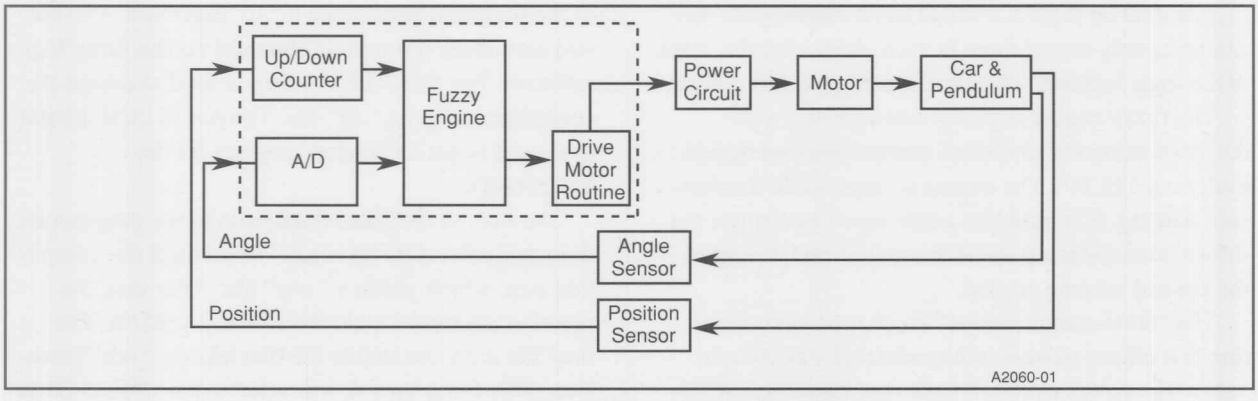


*Figure 1. Block Diagram of the System*

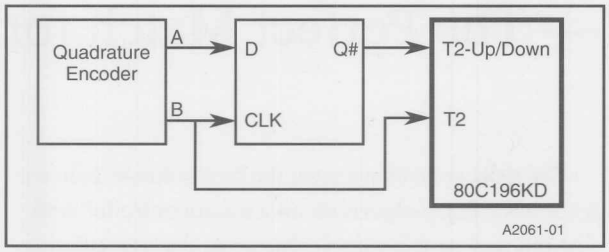


*Figure 2. Connection of the Position Sensor to the 80C196KD*

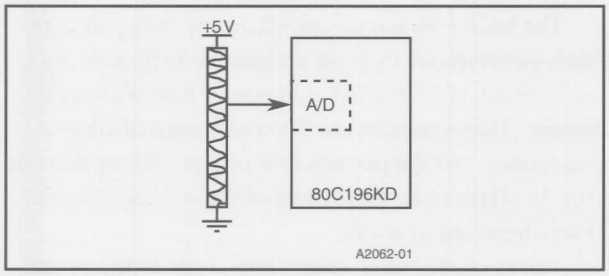


*Figure 3. Angle Sensor Connections*

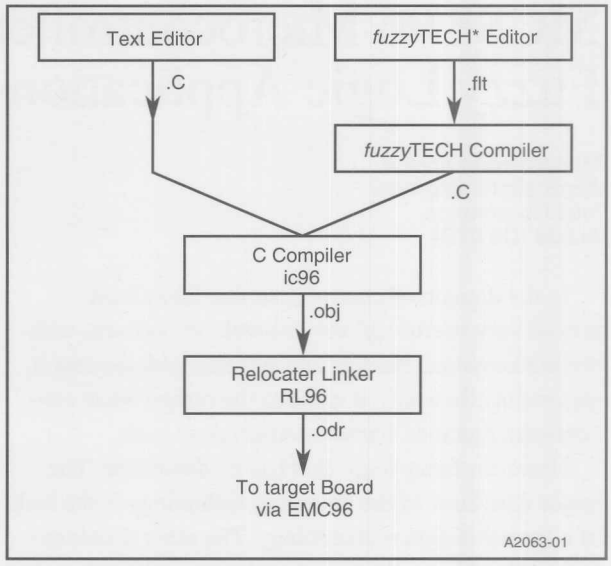


*Figure 4. Program development*

*Continued from page 9*

wave signals. To determine in what direction the shaft is turning, a D flip-flop was used. It outputs a signal that is either high or low, depending on the direction. The electrical connections are shown in Figure 2.

To acquire the distance information, the pulses out of channel B are fed to timer 2 on the 80C196KD controller. Timer 2 is used in the up-down mode. When the T2-up/down pin of the 80C196KD is low, timer 2 counts up; when it is high, timer 2 counts down. Note that the T2-up/down pin is fed the signal from the Q# output of the flip-flop, which allows timer 2 to count up or down depending on the direction.

The sensor used to acquire angle values is simply a potentiometer powered between zero and five volts. When the pendulum swings, it causes the shaft of the sensor to turn. In other words, it moves the location of the cursor of the potentiometer.

The electrical connections between the sensor and the 80C196KD can be represented as Figure 3 shows.

The A/D on the 80C196KD has 8 channels. In this example, only one of them is used. Acquiring the value of the angle is simply a matter of reading the A/D register.

The fuzzy engine is purely software and was designed using the graphical, computer-aided design tool, *fuzzy*TECH*. The engine is responsible for decision making. It is based on a rule base that depicts the information gathered about the system and describes the control scheme needed.

The "drive-motor routine" block represents a routine that allows pulse-width modulated signals to be sent to the power circuit. It uses two registers, PWM1-control and PWM2-control, of the 80C196KD. Writing a number between zero and 255 to these registers causes the pins, PWM1 and PWM2, to output pulse trains of which the duty cycle is dependent on the value that's in the PWMx-control register. The hardware connections were made so that PWM1 drives the car-pendulum system to the left and PWM2 drives it to the right.

The power circuit block is an H-bridge built using discrete components. The motor used is a simple DC motor used in this project at 6.5 volts with a maximum current of 0.4 amps.

The software side of this project consists of an executable program that is made of three major parts: the data acquisition section, the fuzzy controller, and the data ouput section. The program is developed in a PC environment and then downloaded to the 196KD-20 target board on which the 80C196KD resides. (This same target board is supplied with the Project Builder 196 kits.) In order to obtain the executable program, the steps summarized in Figure 4 were taken.

A text editor is used to write the first and third parts of the program, which take a ".C" extension. The second part of the program is obtained via the fuzzy logic software. The *fuzzy*TECH editor is used to design the controller and get a ".ftl" file. The *fuzzy*TECH compiler is used to get a C-coded program for the 80C196KD.

The first, second, and third parts of the program are then merged and iC-96 is used to perform the compilation step, which yields a ".obj" file. After that, the modules are linked and relocated using RL96. The final file is an executable file that takes a ".odr" extension. This file is then downloaded to the 196KD-20 target board using the communication software, ECM96. ■

# Auto Programming the 87C196KD

Jennie Abella
Applications Engineer
Intel Corporation
Article ID# 0705

## Introduction

The Auto Programming mode is a low-cost programming alternative. Using this mode, the contents of an external EPROM are copied to the internal OTPROM of the 87C196KD. This article explains how to construct an inexpensive programmer for the 87C196KD. A programmer for the 27512 is still necessary, but these are readily available. The programming circuit in this article is recommended for programming the 87C196KD and can also be used for the 87C196KC.

## Auto Programming Algorithm

The 87C196KD enters a programming mode when $V_{PP}$ (typically 12.5V) is applied to EA# during the rising edge of RESET#. Then the following sequence is initiated:

1. The upper port 0 pins P0.4–7 (PMODE pins) are read to determine which programming mode is selected. If the value is 1100 (binary), the auto programming mode is entered.
2. The internal OTPROM location 2018H (CCR security lock bits) is checked to see if the security key has been programmed. If these bits are programmed (=0), then the internal security key locations (2020H–202FH) are checked with external locations E020H–E02FH. If security is passed, the program continues. If not, the device enters an endless loop and must be reset to exit it. The LED to indicate that programming has started will not light. (Follow the powerdown procedure and then another attempt can be made to program the device.)
3. The PPW is then loaded from external locations 2014H/2015H, which sets up a 100µs programming pulse. (The equation for this PPW value is shown in "Calculating the Programming Pulse Width" in this article.)
4. PACT# is activated (P2.7=low) to indicate that programming has started. (LED will light.)
5. PVER is initialized (P2.0=high) to indicate that there are no errors to this point. (LED will not be lit.)
6. External data is then read, starting at 4000H.
7. If the word is not 0FFFFH, then the Modified QuickPulse subroutine is called (Step 8). If the word is 0FFFFH, then the word is not programmed and the address is checked to see if programming is completed (Step 10).
8. The PPW timer is started and the data word is written to the OTPROM. The program waits until an interrupt is caused by the PPW timer, which ends the programming pulse. If five writes have not been written, this step is repeated. When five writes have been written to this location, then programming continues with the next word.
9. This location is then read back from the OTPROM and compared to the external location. If it did not program correctly, then PVER is deasserted (P2.0=low) and the LED lights, indicating an error. **Programming will continue even if an error is detected**. The part cannot be programmed again.
10. The address is checked to see if programming has been completed. If the address is 0BFFEH, programming is completed and PACT is deactivated (P2.7=high) and an infinite loop is entered. Continue with the power-down procedure. If programming is not completed, the address pointer is incremented and the next word is programmed.

*Continued on page 12*

*Table 1. Auto Programming Memory Map*

| External EPROM Address | Internal OTPROM Address | Description |
|---|---|---|
| 2014H | N/A | PPW Least-Significant Bit |
| 2015H | N/A | PPW Most-Significant Bit |
| 4000H–BFFFH | 2000H–9FFFH | Reserved locations for code and data. |
| E020H–E02FH | 2020H–202FH | Security key, during verification. |

*Continued from page 11*

## Auto Programming Circuit

Figure 1 is the recommended circuit for auto programming the 87C196KD. Since BUSWIDTH is low, an 8-bit external bus is used. Code to be programmed in the KD at 2000H must be located in the external EPROM starting at location 4000H. The memory remapping is shown in Table 1. The PPW must be located in the external EPROM at location 2014H and 2015H. (See "Calculating the Programming Pulse Width" in this article.)

P0.4–P0.7 are hardwired to $V_{SS}$ and $V_{CC}$ and determine the programming mode. P1.0–P1.2 are used to generate the upper bits of the external EPROM address. The status outputs PACT# and PVER are buffered by a 74HC14 and drive LEDs to indicate Programming ACTive (PACT#) and Programming VERification (PVER). All unused inputs are connected to ground ($V_{SS}$) and unused outputs are left floating. READY, NMI, and BUSWIDTH are active and should be connected as indicated.

Auto programming is specified for a crystal frequency of 6 to 8 MHz. A 27(C)512 EPROM with tACC = 250ns and tOE = 100ns or faster specifications should be used.

## Power-Up and Power-Down Sequencing

**Important:** To avoid damaging the 87C196KD during auto programming, follow these rules:

## Power-Up Sequence

1. When first powering up, the device must be held in reset while $V_{CC}$ stabilizes. $V_{PP}$ and EA# must be allowed to float during this time.
2. Continue holding the device in reset after $V_{CC}$ has stabilized, and apply +12.5 volts to EA# and $V_{PP}$. Refer to the data sheet for exact specifications on this voltage.

   **Warning:** Applying voltage to $V_{PP}$ when $V_{CC}$ is low will permanently damage the device. The recommended circuit shows a switch that interlocks the $V_{PP}$ switch to help prevent damage to the device.
3. Allow time for EA# and $V_{PP}$ to be within tolerance and for the oscillator to stabilize.
4. After condition 3 is met, RESET# may be allowed to rise.

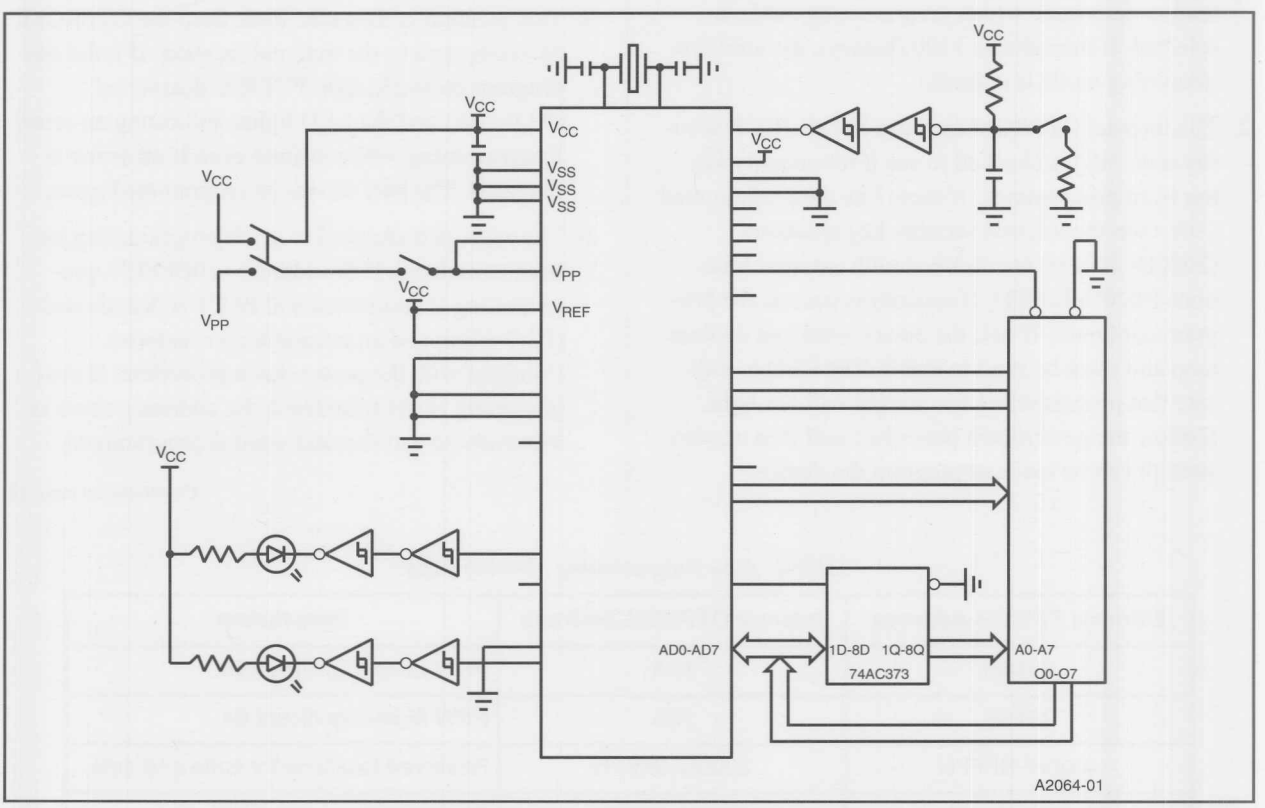


*Figure 1. Auto Programming Mode Circuit*

5. As soon as RESET# rises, the auto programming sequence begins and the PACT# LED turns on.
6. After completion of the auto programming sequence, the PACT# LED turns off and the power-down sequence should be followed.

## Power-Down Sequence

1. Assert the RESET# signal (RESET#=0). RESET# must be held low throughout the powerdown sequence. Do not allow the reset signal to "bounce," or another programming sequence will begin.
2. Remove the +12.5 volts applied to EA# and $V_{PP}$ and allow these pins to float.

   **Warning:** EA# and Vpp must be allowed to float before removing Vcc or the device will be damaged.
3. Turn off the Vcc supply and allow time for this to reach 0 volts.
4. The device can now be removed from the auto programming circuit.

## Calculating the Programming Pulse Width

The programming pulse width needs to be 100μs for the device to program correctly. Use the following formula to calculate the PPW_VALUE. Round the PPW_VALUE to the next higher integer value and load this value in the PPW location in the external EPROM.

$$PPW\_VALUE = (0.6944 \times XTAL1) - 1$$

Where PPW_VALUE is a 15-bit word and XTAL1 is the crystal frequency in MHz.

For example, with an oscillator of 8MHz,

$$PPW\_VALUE = (0.6944 \times 8) - 1$$
$$= 5.55 - 1$$
$$= 4.55$$

This would be rounded to 5. The MSB must always equal 1 (see Figure 2); therefore, for this example, location 2014H/2015H = 8005H.

*Note: External EPROM location 2014H is loaded with the LSB and 2015H is loaded with the MSB of PPW_VALUE.*

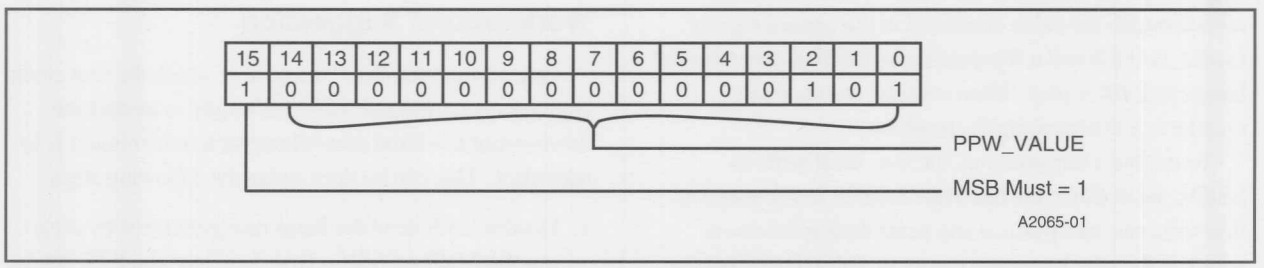


*Figure 2. PPW (2014H/2015H) Example*

# Understanding SSIO Clock and Data Transmissions

Dave Boehmer
Applications Engineer
Intel Corporation
Article ID# 0706

An interesting applications question recently came in that deals with the way the SSIO works. The user observed that the time from data first being placed on the SDx pin to the first clock edge on the SCx pin (data set-up time) varied from transmission to transmission. Since the most-significant bit (MSB) is not changed until the falling edge of SCx, the variable-width MSB resulted in what appeared to be a "delayed clock" as shown in Figure 1.

In order for an SSIO transmission to occur, the SCx and SDx pins must be configured properly and several other conditions must be true. First, the STE bit in the SSIOx_CON register must be set (enabled). Second, the MSB of the SSIO_BAUD register must be set (enabled) and a valid baud rate value must be programmed into the other seven bits. Finally, the user must write the byte to be transmitted to the SSIOx_BUF register — this is what actually starts the transmission.

By writing a seven-bit value to SSIO_BAUD, the user supplies a count from which the baud rate counter will begin its down-count. On each underflow, the counter is reloaded with the value contained in the control register (SSIO_BAUD) and a flip-flop is clocked that generates the baud clock (SCx pin). When enabled, the baud rate counter is a free-running decrementer.

To initiate a transmission, the user must write to SSIOx_BUF. Since the baud rate counter is free-running, this write can take place at any point during the down-count. If the user happens to write to SSIO_BAUD early in the down-count, the first clock edge will not occur until the first underflow (a long delay). This makes the MSB bit time appear relatively long. If the user should write to SSIO_BAUD late in the down-count, the first clock edge will again not occur until the first underflow (a short delay in this case). This makes the MSB bit time appear relatively short. The "variable MSB" is due to the MSB being driven at SDx asynchronously with respect to the baud count underflow.

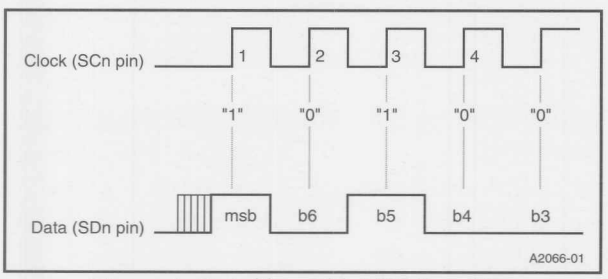


*Figure 1. Variable-width MSB Resulting in "Delayed Clock"*

For example, consider that the user writes 93H (MSB = enabled, other seven bits = 100 KHz) to SSIO_BAUD. The baud count will start counting down from 13H. If the user writes to SSIOx_BUF when the counter is at 11H, the peripheral will place the MSB on the SDx pin and wait until the counter underflow occurs (generating the first clock pulse). It will take approximately 11H baud rate counter ticks (8.5 µs) before the first clock edge is generated. This results in a delayed clock and the MSB being present upon SDx for a relatively long period of time.

If the user writes to SSIOx_BUF when the counter is at 03H, the peripheral will place the MSB on the SDx pin and wait until the counter underflow occurs approximately 3 baud rate counter ticks (1.5 µs) later. This results in the MSB being present upon SDx for a relatively short period of time.

This "variable MSB" (or "delayed clock") condition will occur only for the MSB. Once the MSB is clocked out, the second through eighth (LSB) bits are clocked out consistently at the chosen frequency.

## Work-around Suggestion

One way to "stabilize" the time at which the first clock edge occurs (consistent MSB bit length) is to start the down-count at a fixed time whenever a transmission is to take place. This can be done using the following steps:

1. Disable and clear the baud rate generator by clearing the MSB of SSIO_BAUD. Clear the STE bit in SSIOx_CON to disable transfers.
2. Write the byte to be transmitted to SSIOx_BUF.
3. Set the STE bit in SSIOx_CON to enable a transfer. This will drive the MSB onto the data pin.
4. Enable (start) the baud rate generator by setting the MSB of SSIO_BAUD along with writing the appropriate baud rate value.
5. Rewrite the byte to be transmitted to SSIOx_BUF. This will start the transmission.

Using this procedure (rewriting SSIO_BAUD) will result in a MSB set-up time that is consistent due to the clock being started off at a known point prior to each transmission. Interrupts should be disabled during steps 4 and 5 to prevent interrupt servicing time from causing another variable MSB situation.

# True 3V MCS® 51 Controllers Provide Performance and Power Savings

Curt Durrant
Applications Engineer
Intel Corporation
Article ID# 0707

## Introduction

As the market moves toward low voltage and lower power operation of microprocessors and microcontrollers, Intel is completing development of its first general-purpose microcontroller family designed specifically for true 3.0V operation. The 8XL51FX (FC/FB/FA) and the 8XL5X (52/54/58) microcontrollers will be introduced in Q1 as fully compatible 3.0V versions of their popular 5.0V counterparts. In addition, several design enhancements will make the 8XL51FX and 8XL5X even more attractive for those markets requiring low power and low voltage.

## Low Voltage Range

In contrast to many of our competitors' products, which are being designed at 5.0V and derated for low voltage, the 8XL51FX and 8XL5X are being designed with the specific goal of running at low voltages. Consequently, they will be capable of running at the full frequency range of 3.5 to 16 MHz at voltages as low as 2.7V. To make them compatible with notebook and handheld computer products, which have a $V_{CC}$ range of 3.3V ± 0.3V, the upper tolerance of the $V_{CC}$ range has been extended to 3.6V. This range allows these products to comply with both the regulated and unregulated JEDEC low voltage specs.

The low voltage design is being complemented by I/O buffers that provide higher current source and sink capability than those of devices that are simply derated for lower voltage. I/O buffers designed for 5V operation are designed for $I_{OL}$ and $I_{OH}$ specs that provide adequate drive with minimal noise at that operating voltage. When the design is derated to 3.0V or 3.3V nominal operation, the $I_{OL}$ and $I_{OH}$ specs are significantly reduced. Since the 8XL51FX and 8XL5X are being designed specifically for these lower voltages, the buffers are being optimized to provide greater source and sink currents — hence greater flexibility to the system designer.

## EPROM Lock Bits

The 8XL51FX and 8XL5X products will have improved security features resulting from modifications to the lock bit implementation. The three lock bits will no longer be EPROM cells as they currently are in the 5V products. Instead they will be UPROM cells, which are unerasable. This will prevent a potential "pirate" from selectively erasing the lock bits, allowing him access to the user's EPROM array. In addition to the three lock bits, the devices will also have the 64-byte encryption array currently available on all FX core products.

## Reduced Noise

The noise levels on the existing 5V version of the 8XC51FX products could cause some circuits to fail at 3V. Consequently, several techniques are being implemented in the design of the 8XL51FX and 8XL5X to reduce RFI-generated noise as well as to improve low voltage performance.

First, the $V_{SS}$ and $V_{CC}$ rings have been enlarged and are run in metal only. Enlarging the metal lines provides a larger current-carrying channel for supply current while reducing the effect of IR drop. Running $V_{SS}$ and $V_{CC}$ only in metal eliminates the need for internal "vias," which carry the supply current from one layer to another. This technique eliminates the resistive effect of the vias and the bottleneck they cause in the ground or power path.

Second, the package has two additional $V_{SS}$ pins and one additional $V_{CC}$ pin, making a total of four $V_{SS}$ and two $V_{CC}$ pins for each device. These additional $V_{SS}$ and $V_{CC}$ pins are "reserved" pins on the 5V products. Additionally, the $V_{SS}$ infrastructure that is connected to the internal circuitry has been modified such that the I/O drivers are now connected to their own $V_{SS}$ pin. These design changes should help reduce the effects of noise on the internal circuitry and improve low voltage performance.

## Conclusion

This is the first in a succession of products that Intel will be introducing for 3.0V operation. All of the 8XL51FX products (FC, FB, and FA) and all of the 8XL5X products (52, 54, and 58) will be introduced in Q1 of 1994. Each product will be available in 44-lead PLCC and QFP packages (no DIPs). In addition, they will be completely code-compatible with their 5V counterparts, using the same software development tools. Emulators and EPROM programming support will be provided by third party vendors. Watch for these new low power products to be introduced in Q1 of 1994.

# Timing User MCS® 96 Controller 'C' Code

Larry Ferra
Applications Engineer
Intel Corporation
Article ID# 0708

It is often desirable to know how long it takes a particular piece of code to execute. There are several ways to determine this information, and these ways vary in precision and cost. One way is to set a port pin at the beginning of a section of code being timed and clear the same pin at the end of the section of code. Then you can use an oscilloscope connected to the port pin and measure the time between setting and clearing of the pin. This may not be very accurate if the code being evaluated takes a long time to execute. Another way is to use an in-circuit emulator that can do performance analysis. Another method is to use the on-chip timer of the MCS® 96 controller to time the code. The following small piece of 'C' code uses the MCS 96 controller's on-chip timer2 to perform the code timing. The timing performed in this code is accurate to less than 0.1% error.

```
#pragma model(kd)
#pragma code
#pragma db
#pragma ot(3)
#include <d:\lang\include\80c196.h>


/*  definition of ioc3 which is not included in some versions of iC-96 */

volatile register unsigned char ioc3;
#pragma locate (ioc3=0x0c)


register unsigned int time_low,time_high,overhead;

#pragma interrupt(overflow=0)
void overflow(void)         /*   Interrupt routine to catch overflow */
{
 time_high++;
}

#define init_timer()    ioc2=1;ioc1=0x28;int_mask=1;asm ei;
#define time_start()    timer2=0;time_high=0;wsr=1;ioc3=0x01;wsr=0;
#define time_end()      wsr=1;ioc3=0x00;wsr=0;time_low=timer2;
#define calc_time(a)    (*(unsigned int *)((unsigned int)(&a)+2))=time_high;\
                         asm sub a, time_low, overhead;


/*    The following routine calculates the overhead in executing the
code to save the timer value     */
void calc_overhead()
{
 init_timer();
 time_start();
 time_end();
 overhead = time_low;
}

/*   example code illustrating the timing usage   */

/*  The timing result variables must be filescope register variables
for the macros to function properly.  */

 register unsigned long time1,time2;

main()
```

*Figure 1. Code Example*

```
{
 register int x;

 calc_overhead();          /*  figure overhead of saving timer  */

 time_start();             /*  start the timer   */
 for(x=0;x<1000;x++);      /*  execute some code  */
 time_end();               /*  stop timer   */
 calc_time(time1);         /*  calculate time and save in time1  */

 time_start();             /*  start timer again   */
 for(x=0;x<3000;x++);      /*  execute some other code   */
 time_end();               /*  stop timer  */
 calc_time(time2);         /*  calculate time and save in time2  */
}
```

*Figure 1. Code Example (Continued)*

The variables time1 and time2 contain the number of states ($2 / F_{OSC}$) it takes to execute the respective sections of code. For example, if time1 = 20,000 after execution, then this value corresponds to 20,000 x (2 / 20,000,000 at 20 MHz) ns or 2.0 milliseconds to execute.

# How to Choose a Memory Device

Rick Evans
Applications Engineer
Intel Corporation
Article ID# 0709

Each manufacturer uses different nomenclature for AC timings. When working with different vendors' datasheets, it can be confusing translating back and forth between them. This article explains how to choose a memory device with the correct speed to work with one of the MCS® 96 microcontrollers.

Let's choose the 80C196KB, which runs at 16 MHz and has a time-multiplexed address/data bus. That means that the same pins are used to output the address and to input and output the data. Three memory timings are critical when selecting a memory device for the MCS 96 microcontroller: the address access time (tACC), output enable delay (tOE), and data float time (tDF). These three specs can be found in any memory device datasheet. They might have different acronyms, but their descriptions will be the same.

The tACC is the time it takes when applying a stable address for the correct data to appear on the data output pins. Each memory device has an output enable (OE#) pin. This pin is usually connected to the read signal (RD#) of the 80C196KB. The tOE time is the time it takes once this output enable pin is held low for correct data to appear on the output pins. The tDF spec is the time it takes for the memory device to float the data pins after the output enable pin (OE#) is disabled. This spec is important because you want to make sure that the memory device gets the data off the bus before the 80C196KB starts driving the next address. Contention occurs when both the memory device and the 80C196KB are both driving the same bus.

The three specs, tACC, tOE and tDF, correspond to the 80C196KB specs of $T_{AVDV}$ (address valid to data valid), $T^{RLDV}$ (read low to data valid), and $T_{RHDZ}$ (read high to data float). Remember that since the 80C196KB has a multiplexed address/data bus, the address must be latched. The latch delay must be included in the calculations. The relationships between these specs are as follows:

$tACC \leq T_{AVDV}$ + latch

$tOE \leq T_{RLDV}$

$tDF \leq T_{RHDZ}$

Let's take the Intel boot-block Flash 28F200BX-60. Looking up the three important specs, we find the values in Table 1.

*Table 1. Intel Boot Flash 28 F200BX-60 Spec Values*

| | 80C196KB (16 MHz) | |
|---|---|---|
| **28F200BX-60** | **Tosc = 1 / Fosc = 62.5 ns, Latch Delay = 11 ns (given)** | |
| tACC = 70 ns | $T_{AVDV} = 3T_{OSC} - 55$ | $T_{AVDV}$ = 132.5 ns + latch = 147.5 ns |
| tOE = 35 ns | $T_{RLDV} = T_{OSC} - 23$ | $T_{RLDV}$ = 39.5 ns |
| tDF = 25 ns | $T_{RHDZ} = T_{OSC} - 20$ | $T_{RHDZ}$ = 42.5 ns |

Comparing the 80C196KB specs with the Flash specs, we find that we can use this Flash device with the 16 MHz 80C196KB with no wait states. If we were to use a slower Flash device while running at 16 MHz, we would have to insert wait states. Calculating with wait states is just as easy. Just add 2 x Tosc x *n*, where *n* is the number of wait states. This wait state delay value needs to be added to $T_{AVDV}$ and $T_{RLDV}$ but not $T_{RHDZ}$.

Example for calculating with one wait state:

$T_{AVDV}$ (1wait) = 5 $T_{OSC}$ – 55 = 257.5 ns

$T_{RLDV}$ (1 wait) = 3 $T_{OSC}$ – 23 = 164.5 ns

$T_{RHDZ} = T_{OSC} - 20$ = 42.5 ns

Look for more information on wait states in the next issue.

# Low-Cost Tools: Evaluation Boards vs. the 196KD Target Board

Will Schreiber
Senior Program Manager
Intel Corporation
Article ID # 0710

A number of readers have written with questions about the target board mentioned in the "Project Builder 196" cover article in the Q3, 1993, Embedded Applications Journal. Readers were unfamiliar with the term "target board" and requested information about the differences between Intel's evaluation boards and the 196KD target board. This article compares the features

*Continued on page 20*

*Table 1. Target Board and Evaluation Board Features Comparison*

| Features | Evaluation Board | 196KD Target Board |
|---|---|---|
| **Cost** | $ 260.00 – 525.00 | $ 196.00 |
| **Monitor/Debugger** | **ECM** | **DeBug Monitor** |
| O/S | DOS | Windows* 3.1 |
| Input | Command Line | Edited Windows |
| Serial Port Speed | 9600 baud | 9600 – 57.6K baud |
| Monitor Retargeting | No | Yes |
| Debugger Features | (same) | (same) |
| **Board Hardware** | | |
| Power | 5V, ±12V | 5V only |
| RS-232 Ports | 2 | 1 |
| Flash Supported | No (28-Pin Sockets) | Yes (32-pin JEDEC) |
| 8-bit Memory | On Board | On Board |
| 16-bit Memory | On Board (16 bit MPU's only) | Off Board Memory Expansion |
| Add'l Memory Sockets | On Board | Off Board Memory Expansion |
| A/D Input | Input Protected Circuitry | Direct Chip Interface |
| iRISM | External EPROM | Internal Device EPROM |
| Standalone System | Yes | Yes |
| Code Start Address | 0000H (MCS® 51 controller) or 2080H | C000H |
| **Other Product Features:** | | |
| Hardware Manual | Yes | Yes |
| Schematics | Yes — Paper | Yes — OrCAD* files and Paper |
| OrCAD Device Library | No | Yes — 114 Intel Devices |
| iRISM Source | Yes — Paper | Yes — Software File |
| Demo ASM | Yes — With Registration | Yes — In the Box |
| ASM Prog. Manual | Yes — ASCII Text Files | Yes — Hypertext Manual |
| Demo C Compiler | No | Yes — With Registration |
| C Prog. Manual | No | Yes — Hypertext Manual |
| Device Manual | No — (On Intel BBS) | Yes — Hypertext Manual |
| Data Sheets | No — (On Intel BBS) | Yes — Hypertext |
| *Ap*BUILDER | No — (On Intel BBS) | Yes |
| *Model*BUILDER | No | Yes |
| Sample Code | No | Yes — 10 Annotateded Files |

of the target board kit with those of the evaluation board kits and lists the products that are currently available.

With the transition of Intel In-Circuit Emulators and compiler tools to third party vendors, a number of customers asked if Intel would continue to provide low-cost tools for embedded control. Yes, Intel will continue to address reader questions about the availability of these tools, a listing is provided in Table 2.

Intel's FaxBack* documents can be retrieved with a touch-tone phone and sent directly to your fax machine by calling **1-800-628-2283** or **916-356-3105**.

*Table 2. Tools for Embedded Control*

| **Evaluation Board Product Reference** | | | |
|---|---|---|---|
| **Devices Covered** | **Intel Product Order Code** | **List Price** | **FaxBack #** |
| 8097BH | EV8097BH | $ 320.00 | 2036 |
| 51/52/54/58 & FA/FB/FC | EV80C51FX | $ 260.00 | 2041 |
| 51GB | EV80C51GX | $ 290.00 | 2038 |
| 186EA/XL | EV80C186EAXL | $ 342.00 | 2084 |
| 186EB | EV80C186EB | $ 395.00 | 2085 |
| 186EC | EV80C186EC | $ 450.00 | 2143 |
| 196KB | EV80C196KB | $ 320.00 | 2039 |
| 196KC | EV80C196KC | $ 352.00 | 2034 |
| 196KD | EV80C196KD | $ 395.00 | — |
| 196KR/KT | EV80C196KR | $ 390.00 | 2040 |
| 196MC | EV80C196MC | $ 495.00 | 2551 |
| 196MD | EV80C196MD | $ 525.00 | — |
| **Board Kits with Software:** | | | |
| **Features** | **Intel Product Order Code** | **List Price** | **FaxBack #** |
| Project Builder 196 | PROJBLD196KBCD | $ 196.00 | 2643 |
| Paradigm Software | DK80C186EAXL | $ 582.00 | 2086 |
| Paradigm Software | DK80C186EB | $ 632.00 | 2086 |
| Paradigm Software | DK80C186EC | $ 692.00 | 2086 |
| Microsoft & SSI | DKCVC186EAXL | $ 1380.00 | — |
| Microsoft & SSI | DKCV80C186EB | $ 1450.00 | — |
| Microsoft & SSI | DKCV80C186EC | $ 1510.00 | — |
| **Project Builder 196 Kits with Fuzzy Logic Software (All of these kits include Project Builder 196)** | | | |
| **Features** | **Intel Product Order Code** | **List Price** | **FaxBack #** |
| Explorer 96 | FUZZYBLDREXP96 | $ 396.00 | 2174 |
| MCU-96 Edition | FUZZYBLDRMCU96 | $ 2096.00 | 2174 |
| MCU-96 & NeuroFuzzy Module | FUZZYBLDRNFK96 | $ 2996.00 | 2174 |
| MCU-96 & Real-Time Remote Cross Debugger | FUZZYBLDRXDB96 | $ 4896.00 | 2174 |

# New Embedded Control FaxBack® Service Documents

**New/Updated as of November 15, 1993**

| Title | Number |
|---|---|
| iPLD610/910 Change Notification | 2537 |
| Catalog of Technical Information Not in Data Books | 2647 |
| iPLD: Competitor Cross Reference Guide | 2059 |
| MCS-96: BMOVI Instruction | 2058 |
| 80960CF: Technical Bulletin Number 5.0 | 2180 |
| PCMCIA Card Drivers & Reader/Writers - Vendor and Condensed Product Description Lists | 2201 |
| PCMCIA Card Drivers & Reader/Writers - Detailed Technical Specifications | 2221 |
| 80960CA: Technical Bulletin #12 | 2181 |
| 80960CA/CF: Microprocessor Self Test Details | 2182 |
| Flash Memory Socket Adapters | 2574 |
| 8XC196KB History and Errata (MC1293) | 2548 |
| 28F016SA: Programming Support | 2213 |
| 28F032SA: Programming Support | 2222 |
| iFX780/iFX740: Product Brief for "PROTAG" | 2066 |

**New/Updated as of October 27, 1993**

| Title | Number |
|---|---|
| InfoGuide: 28F032SA 32-Mb FlashFile™ Memory | 2722 |
| InfoGuide: 28F016SA 16-Mb FlashFile Memory | 2718 |
| InfoGuide: 28F001BX 1-Mb Boot-Block Flash Memory | 2717 |
| InfoGuide: Intel Flash Drives (5 MB, 10MB) | 2721 |
| InfoGuide: Series 2+ Flash Memory Cards (4 MB, 20 MB, 40MB) | 2719 |
| Flash Memory Quick Reference Guide | 2216 |
| Book Abstract: Designing with Flash Memory | 2220 |
| Flash Memory Technical Support Summary | 2204 |
| Flash Memory Socket Adaptors | 2574 |
| Flash Memory Card Programming Support | 2200 |
| Flash Memory Card Drives, Card Reader/Writers Support | 2201 |
| BIOS Vendor Contacts for PCMICA-ATA-IDE Flash Drives | 2215 |
| 28F016SA: Technical Support Summary | 2212 |
| 28F016SA: Programming Support | 2213 |
| 28F016SA: Software Drivers on BBS | 2214 |
| 28F008SA: Programming Support | 2582 |
| 28F400BX, 28F004BX: Technical Support Summary | 2520 |
| 28F400BX, 28F004BX: Programming Support | 2581 |
| 28F200BX, 28F002BX: Technical Support Summary | 2521 |
| 28F200BX, 28F002BX: Programming Support | 2580 |
| 28F001BX: Programming Support | 2579 |
| 28F020: Programming Support | 2578 |
| 28F010: Programming Support | 2577 |
| 28F512: Programming Support | 2576 |
| 28F256A: Programming Support | 2575 |
| Programmer Vendor Address & Phone Numbers | 2203 |
| Intel's Flash Memory Boot Block Architecture for Safe Firmware Updates | 2218 |

# Q's & A's

Article ID # 0711

## MCS® 96 Microcontroller Family Q's & A's

**Q: When using a ROM, EPROM, or OTP 8XC196 part and running exclusively from internal memory (i.e., making no external bus accesses), what happens to the address/data bus?**

A: The pins are high impedance; therefore, the bus can be used by external devices or ports 3 and 4 can be used as standard I/O pins.

**Q: Can the auto programming circuit for the 87C196KC, which uses the upper address lines of the 87C196KC to address the external EPROM, also be used for the 87C196KD?**

A: No. For an 87C196KD auto programming circuit, use the circuit described in "Auto-programming the 8XC196KD" in this journal. It uses Port 1.0–2 to generate the upper address bits of the external EPROM. This circuit can also be used to program the 87C196KC.

**Q: I am trying to use the timers of the 8XC196 device's HSO to generate periodic interrupts. How does this work?**

A: It is typical to start a software timer using T1, since it is free running, and have a SWT interrupt routine enter another event in the CAM (without locking) with hso_time = timer1 + time_offset. This allows the T1 timer to be used for other events, and T2 can still be used as a counter.

## Evaluation Board Q's & A's

**Q: Where can I get the source code for the RISM-XX on my evaluation board?**

A: It comes on the disk supplied with the board and is also available from the Intel BBS.

**Q: Why does the 80C196KB, KC, KD, KR, MC evaluation board decode WRL and WRH strobes?**

A: The EV80C196KB, KC, KD, KR, MC evaluation boards can write to byte and word locations, so AD0 and BHE# lines must be decoded to generate the needed WRL# and WRH# signals for odd and even read and write cycles.

**Q: Is ASM-XX the only assembler that works with the evaluation boards?**

A: We recommend that you use ASM-XX or RL-XX, Intel C, iC-96, or other Intel-compatible programming languages. Be aware that not all assemblers and compilers produce Intel-compatible object code. If your assembler or compiler produces Intellec Hex output, the HEXOBJ.EXE program (supplied with the evaluation board software and available on the Intel BBS) will convert the hex file to an Intel OMF file format that can be downloaded onto an evaluation board.

**Q: I used the OH.EXE program on my disk to convert my ASM-51 object file to Intel HEX format, but have not been able to download it to the evaluation board successfully. What do you suggest?**

A: When using Intel tools like ASM-51 or ASM-96 there is no need to convert your object files, since they are already in the correct format for downloading. If using another brand of assembler or compiler, there may be a need to convert the file to make it Intel-compatible.

**Q: If a HEXOBJ.EXE file does not exist for my code developed with tools from another company, how can I get it to download to the evaluation board?**

A: Nearly all third party vendors produce Intel-compatible hex files. Therefore, the Intel utility, HEXOBJ.EXE, should work to convert the hex file to Intel OMF object files. If not, you must take this issue to your assembler/compiler manufacturer. For MCS® 96 and MCS 51 microcontrollers, the HEXOBJ.EXE file is available on the Intel BBS.

**Q: What can I do with the single line assembler of the evaluation board 80CXXXXX?**

A: It allows you to enter RUN environment and to step through the program flow and make minor changes/patches to system software under test.

**Q: I am having wait state problems with an 80C196KB,KC,KD, or 8097BH type board. What could be wrong?**

A: Ensure that Jumper E21 on these boards is set in the correct position, since the boards are typically shipped with wait states disabled. Refer to the user's manual for settings.

## Project Builder 196 Q's & A's

**Q: What microcontrollers does Project Builder 196 support?**

A: Currently the kit is shipped with an 87C196KD-20 microcontroller. Because the KD is a superset of the KC and KB, if you know the differences, you can use the 196KD target board to develop code that can be used in your 196KB/KC applications.

**Q: When I try to use DASM, I get an error, "Invalid Opcode at xxxxH." What's causing it?**

A: Your memory is probably jumpered incorrectly. Refer to the list of memory jumper configurations on page B-2 of the *196KD-20 Microcontroller Target Board User's Manual.*

**Q: The SRAM device that's on my board is not listed in the memory configuration table on page B-2 of the user's manual. Which SRAM configuration should I use?**

A: Many boards were sent out with a different SRAM than those listed in the manual. If your SRAM has part number 81C78A-35P, use the configuration for part number dt7164 (8 Kbytes).

**Q: What are the differences between the Project Builder 196 target board and an evaluation board?**

A: The evaluation board is shipped with a larger memory (24K), but the Project Builder 196 target board can be modified to accommodate up to 128K of onboard memory. Also, when communicating with your PC, the evaluation board's serial port is available for use, but the target board's serial ports are tied up. However, when operating in standalone mode, the serial port is available on both the target board and the evaluation board. Other minor differences are shown in the following list:

### Target Board

Address lines A0-A7 are unlatched, so the address/data lines remain multiplexed.

No onboard wait state control.

Simple RC reset circuitry.

Requires only +5V and ground.

Monitor software provides user-friendly Windows* software-based interface.

### Evaluation Board

Address lines are latched, so the address and data lines are brought out separately.

Onboard PAL controls waitstates.

Schmitt trigger-controlled reset.

Requires +12V, –12V, +5V and ground.

Evaluation board interface uses ECM software in DOS.

## *Corporate Distributors*

Please use these numbers to be directed to your local EMD distributor.

| | |
|---|---|
| Alliance Electronics | 505-292-3360 |
| Anthem Electronics | 408-453-1200 |
| Arrow/Schweber Electronics | 800-777-ARROW |
| Hamilton Hallmark | 800-888-9236 |
| Pioneer Standard | 216-587-3600 |
| Pioneer Technologies Group | 301-921-0660 |
| Wyle Laboratories | 714-753-9953 |
| Zentronics | 416-507-2600 |
| Zeus | 800-52-HIREL |

## BBS adds new area for *Ap*BUILDER and Hypertext

The BBS has a new area level (25) just for *Ap*BUILDER files and hypertext manuals and data sheets. Area level 25 has four sublevels: (1) General, (2) 196 Files, (3) 186 Files, and (4) 8051 Files.

Look for *Ap*BUILDER files in the General sublevel. Look for hypertext manuals and data sheets in the appropriate product-family sublevel. Remember that the latest revisions are always available first from the BBS.

# Errata and Change Identifiers

Article ID #0611

Change identifiers have been used since 1990 to distinguish revisions, or steppings, of embedded control devices. Older devices have no change identifiers. On most devices, the change identifier is the last character in the FPO number, which is typically a nine-character code on the second line on the top of the device. An example FPO number is "L1234567D," in which "D" is the change identifier. On some devices, such as the 8XC51SL-BG, the change identifier is a separate line item and uses several characters. For example, change identifier "SW011" identifies the B-3 stepping of the 8XC51SL-BG.

This article lists change identifiers, errata, and design considerations for recent steppings of embedded control products. For many of these devices, complete errata listings or explanations are available from the FaxBack* service. The "Ref" column in each table lists FaxBack service document numbers for errata lists and explanations, and "For More Information" at the end of this article has a complete list of related document numbers and titles.

## MCS® 51 Microcontroller Family Errata and Design Considerations

This list covers the most recent steppings of the MCS® 51 microcontrollers. A complete list of the errata for all steppings is available on the FaxBack service (document #2632).

*Table 1. MCS® 51 Microcontroller Family Errata and Design Considerations*

| Device | Step | Change Identifier | Errata and Design Considerations | Ref. |
|---|---|---|---|---|
| 8051AH/8031AH | C | A | 1. External interrupt 0 errata | 2154<br>2161 |
| | C-3 | B | No known errata | |
| 80C51BH/80C31BH | C | none | 1. Reset lockup problem<br>2. High IPD if C does not equal B.7 before going into powerdown<br>3. ROM verify mode fails<br>4. Steam passivation problem on plastic parts | |
| | C-1 | none | 1. High IPD if C does not equal B.7 before going into powerdown<br>2. ROM verify mode fails | |
| | D | D or 2 | No known errata | |
| 87C51 | D | A | No known errata | 2106 |
| 80C52/80C32 | C | none | 1. RST/ONCE mode problem | |
| | A | A | No known errata | |
| 83C51FA/80C51FA | C | none | 1. PCA errata | 2528 |
| 87C51FA | C | none | 1. RST/ONCE mode problem<br>2. PCA errata | 2528 |
| | D | A | No known errata | 2107 |
| 8XC51FB | A | none | 1. PCA errata | 2528 |
| | B | A | No known errata | 2111 |

*Table 1. MCS® 51 Microcontroller Family Errata and Design Considerations (Continued)*

| Device | Step | Change Identifier | Errata and Design Considerations | Ref. |
|---|---|---|---|---|
| 8XC51FC | A | none | 1. Port 1, 2, 3 problem — asychronous port reset not supported<br>2. Failed ESD qual testing | 2028 |
| | B | none | No known errata | |
| 8XC51GB | B | none | 1. Reset polarity changed to active low<br>2. Port 1 reset value changed to all zeros | 2032<br>2032 |
| | B-2 | none | No known errata | |
| 8XC152JX | B | none | 1. AE/RDN race condition<br>2. Receive FIFO is not cleared when receiver is enabled<br>3. DMA errata<br>4. SDLC flag recognition errata | <br>2030<br>2118<br>2035 |
| | C | none | No known errata | |
| 8XC51SL-BG | B-3 | SW011 | 1. GATEA20, RCL hardware speedup processing<br>2. Powerdown current stabilization<br>3. Port 2 address mux<br>4. KSI powerdown wakeup interrupt<br>5. Reset errata | 2008<br><br><br><br>2114 |
| | B-4 | SW062 | 1. GATEA20, RCL hardware speedup processing<br>2. Powerdown current stabilization<br>3. Port 2 address mux<br>4. KSI powerdown wakeup interrupt | 2008 |
| 8XC51SL-AH/AL | A-0<br>A-1<br>A-2 | AA<br>BA<br>CA | 1. Power-down current stabilization<br>2. System power management errata | 2048 |

## MCS® 96 Microcontroller Family Errata and Design Considerations

This list covers the most recent steppings of the MCS® 96 microcontroller. Complete lists of the errata for all steppings are available on the FaxBack service for the 8X9XBH (#2134), the 8XC196KB (#2548), the 8XC196KC (#2136), the 8XC196KR (#2527), and the 8XC196NT (#2178).

*Table 2. MCS® 96 Microcontroller Family Errata and Design Considerations*

| Device | Step | Change Identifier | Errata and Design Considerations | Ref. |
|---|---|---|---|---|
| 8X9XBH | D | D | 1. Indexed 3-operand multiply<br>2. HSI FIFO<br>3. Reserved location 2019H<br>4. RESET and the QBD pins<br>5. Software RESET timing<br>6. Using T2CLK for Timer2 | 2134 |
| | E | E | 1. Indexed 3-operand multiply<br>2. HSI resolution<br>3. Reserved location 2019H<br>4. Reserved location 201CH | <br><br>2631<br>2140 |

*Table 2. MCS® 96 Microcontroller Family Errata and Design Considerations (Continued)*

| Device | Step | Change Identifier | Errata and Design Considerations | Ref. |
|---|---|---|---|---|
| 8XC196KB<br>8XC196KB10/KB12 | B | B | 1. Divide during HOLD or READY<br>2. HSI 8/9 state<br>3. SIO framing error<br>4. SIO RI flag<br>5. DJNZW instruction<br>6. CMPL with R0<br>7. ALE glitch | 2548<br><br><br><br><br>2156<br>2568 |
| 8XC196KB/KB16 | B | D | 1. Divide during HOLD or READY<br>2. HSI 8/9 state<br>3. CMPL with R0<br>4. Missed EXTINT P0.7 | 2122<br><br>2156<br>2049 |
| | C | E | 1. HSI 8/9 state<br>2. CMPL with R0<br>3. Missed EXTINT P0.7 | <br>2156<br>2049 |
| 8XC196KC | | | The number following each entry in this list is a cross-reference to the applicable section of FaxBack service document #2136. | 2136 |
| | all | | Design Considerations<br>1. Indirect shift count value 116<br>2. Write cycle during Reset 147 | |
| | A | A or none | **Errata**<br>1. A/D convert error 101<br>2. BMOVI instruction 105<br>3. Divide error during hold 109<br>4. HSO IOC1 bits interchanged 115<br>5. Port 0 latched on wrong phase 126<br>6. PTS Req during INT latency 131<br>7. NMI during PTS latency 132<br>8. SIO Mode 0 140<br>9. TIJMP INDEX_MASK value 143<br>10. Serial Port Framing Error 150<br>11. QBD glitch during powerup 163<br>12. A/D linearity too large 173<br>13. Analog input latch-up 174<br>14. INST weak during CCB fetch 175<br>15. 2 CCB fetches 176<br>16. 3 wait states during CCB 177<br>17. Pullups too weak during Reset 178<br>18. Buswidth always 16 bits 179<br>19. Vcc glitch resets device 180<br>20. Incomplete reset at > 12 MHz 181<br>21. Oscillator startup 215<br>22. Reset hysteresis 216<br>23. Missed EXTINT P0.7 2049 | |
| | B-1 | B | 1. Divide error during hold 109<br>2. NMI during PTS skips address 123<br>3. QBD glitch during powerup 163<br>4. ONCE mode entry 214<br>5. Oscillator startup 215<br>6. Reset hysteresis 216<br>7. Missed EXTINT P0.7 | 2049 |
| | B-3 | D or E | 1. Divide error during hold 109 | |

*Table 2. MCS® 96 Microcontroller Family Errata and Design Considerations (Continued)*

| Device | Step | Change Identifier | Errata and Design Considerations | Ref. |
|---|---|---|---|---|
| 8XC196KC (Continued) | | | 2. NMI during PTS skips address 123<br>3. QBD glitch during powerup 163<br>4. Reset hysteresis 216<br>5. Missed EXTINT P0.7 | 2049 |
| 8XC196KD | A-1 | B | 1. Missed EXTINT P0.7 | 2049 |
| 8XC196KR/JR/KQ/JQ | A, C | A, C | Design Considerations<br>1. P6_REG not updated immediately<br>2. Write cycle during reset<br>3. EPA timer reset/write conflict<br>4. Valid time matches<br>5. CLKOUT<br>6. Indirect shift operation<br>7. Internal RAM powerdown leakage<br>8. A/D latchup<br>9. INST operation<br>10. KQ/JQ memory map | 2527 |
| | A | A | **Errata**<br>1. Oscillator noise sensitivity<br>2. Slave programming mode<br>3. A/D abort<br>4. PTS with other interrupts<br>5. PTS/NMI conflict<br>6. Data output register cleared when mode register is written<br>7. Divide error during Hold/Ready<br>8. SIO Mode 0<br>9. Remap mode on EPA3<br>10. Serial port framing error<br>11. EPAIPV value multiplied by 2<br>12. EPA_MASK1/EPA_PEND1 must be written as words<br>13. Interruptable block move (BMOVI) | |
| | A, C | A, C | 1. Ioh2 = – 6 µA | |
| | C,D | C,D | 1. Cannot access external locations 1B00H-1BDFH | |
| 8XC196NT | D | D | Design Considerations<br>In bus controller modes 1 and 2, in 8-bit bus mode, the upper address lines need to be latched | |

## 8XC186/8XC188 Family Errata and Design Considerations

*Table 3. 8XC186/8XC188 Family Errata and Design Considerations*

| Device | Step | Change Identifier | Errata and Design Considerations | Ref. |
|---|---|---|---|---|
| 80C186A, B | none | | 1. Non-contiguous Interrupt Acknowledge cycles<br>2. ERROR# processing during FWAIT instructions<br>3. Input high voltage requirement on SRDY and ARDY pins<br>4. Interrupt Status Register (DHLT and Timer Interrupts)<br>5. Bus preemption errata (HOLD/HLDA protocol)<br>6. 80C188 RFSH# pin output timing | 2096 |
| 80C186XL | A | none | Never put into production | |
| | B | A | 1. INTx/INTAx in Cascade Mode | 2025 |
| | C | B | No known errata | |
| 80C186EA/80L186EA | A | A | 1. Low hysteresis on RESIN# pin<br>2. TEST/BUSY#, RD#/QSMD#, LCS#, and UCS# input low voltage<br>3. INTx/INTAx in Cascade Mode | 2025 |
| | B | B | 1. INTx/INTAx in Cascade Mode | 2025 |
| 80C186EB/80L186EB | A | A or none | 1. Entry into ONCE mode<br>2. Low hysteresis on RESIN# pin<br>3. SINT1 input not latched internally<br>4. Ready input during INTA# bus cycle<br>5. CLKOUT transitions on the rising of CLKIN instead of the falling edge<br>6. I/O ports 1 and 2 initialize to Port instead of Peripheral function (documentation error)<br>7. INTx/INTAx in Cascade mode | 2025 |
| | B | B | 1. INTx/INTAx in Cascade Mode | 2025 |
| 80C186EC | A | A | 1. Early exit from Reset (with high Vcc, or low temperature, or both)<br>2. Powersave Mode initialization at Reset | |
| | B | B | No known errata | |

# For More Information

The following FaxBack* service documents contain errata lists and explanations.

**MCS® 51 Microcontroller Family**

| Title | Number |
|---|---|
| 8051/31AH Shrink C-Step External Interrupt 0 Errata | 2161 |
| 8051/31AH Shrink Conversion | 2154 |
| 80C31/51BH D-Step Marking | 2015 |
| 80C51SL-BG Errata | 2130 |
| 80C51SL-BG Errata, Version 2.6 | 2008 |
| 80C51SL-BG Product Preview Data Sheet Errata | 2630 |
| 80C51SL-BG Reset Errata | 2144 |
| 87C51 D Step | 2106 |
| 87C51FA D Step | 2107 |
| 87C51FB/83C51FB B Step | 2111 |
| 87C51FC Data Sheet Errata I | 2024 |
| 87C51FC Data Sheet Errata II | 2020 |
| 87C51FC Port Anomaly | 2028 |
| 87C51GB A-1 & B Errata & Design Considerations | 2032 |
| 87C54/80C54 Data Sheet Errata I | 2022 |
| 87C54/80C54 Data Sheet Errata II | 2021 |
| 8XC152 DMA Bug | 2118 |
| 8XC152 Global Serial Channel Bug | 2030 |
| 8XC152 SDLC Flag Recognition Bug | 2035 |
| 8XC51FA/FB/FC Hardware Description Errata | 2017 |
| 8XC51FX PCA/Timer2 Errata | 2528 |
| MCS® 51: 1992 8-bit Datasheet Errata | 2165 |

**MCS® 96 Microcontroller Family**

| Title | Number |
|---|---|
| 8X97BH, 8X97JF Reserved Memory Location 2019H | 2631 |
| 8X9X Reading 201CH Bug | 2140 |
| 8X9X SIO Status Register Errata | 2138 |
| 8X9XBH, 8X9XJF, and 8X98 Bug History | 2119 |
| 8X9XBH, 8X9XJF, and 8X98 Conversions | 2133 |
| 8X9XBH, 8X9XJF, and Bug List | 2134 |
| 80C196 Oscillator Noise Sensitivity | 2113 |
| 8XC196 Hold/Ready DIV, DIVB Problem | 2122 |
| 8XC196KB and 8XC198 Bug List | 2135 |
| 8XC196KB History and Errata | 2548 |
| 8XC196KB ALE Glitch | 2568 |
| 8XC196KB/198/194 CMPL Instruction Using R0 | 2156 |
| 8XC196KC 1991 Handbook Errata | 2131 |
| 8XC196KC Bug List | 2136 |

**MCS® 96 Microcontroller Family (Continued)**

| Title | Number |
|---|---|
| 8XC196KC HSI PTS Handbook Errata | 2141 |
| 8XC196KC/KD 1992 User's Manual Errata | 2570 |
| 8XC196KC: NMI Iih1 Specification Change | 2164 |
| 8XC196KR Errata and Design Considerations | 2125 |
| 8XC196KR/JR/KQ/JQ Errata | 2527 |
| 8XC196NT Errata | 2178 |
| 8XC198 and 8XC196KB/KB16 Data Sheet Errata | 2569 |
| Itl Specification Change on 8XC198, 8XC196KB/8XC196KB16 | 2567 |

**80C186/80C188 Microcontroller Family**

| Title | Number |
|---|---|
| 186 Design Spurious Writes | 2553 |
| 186 Family Eval Board Errata | 2592 |
| 186 Family User's Manual Errata | 2603 |
| 80C186/188 Bus Preemption Problem | 2096 |
| 80C186/80C188 Interrupt Controller Error | 2025 |
| 80C186/C188 B-Step Technical Bulletin | 2100 |
| 80C186/C188EA A-Step Technical Bulletin | 2102 |
| 80C186/C188EB A-Step Technical Bulletin | 2103 |
| 80C186/C188EB B-Step Technical Bulletin | 2104 |
| 80C186/C188EC A-Step Technical Bulletin | 2105 |
| 80C186/C188XL B-Step Technical Bulletin | 2101 |
| 80C18x XL/EA/EB INTx/INTAx# Errata | 2025 |
| EV80C186EB Eval Board Manual Errata | 2158 |

# Sales Offices and Distributor Addresses


Intel Corporation
Literature Department
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
United States

Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi
Ibaraki, 300-26
Japan

Intel Corporation S.A.R.L.
1, Rue Edison, BP 303
78054 Saint-Quentin-en-Yvelines Cedex
France

Intel Corporation (U.K.) Ltd.
Pipers Way
Swindon
Wiltshire, England SN3 1RJ
United Kingdom

Intel GmbH
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen
Germany

Intel Semiconductor Ltd.
32/F Two Pacific Place
88 Queensway, Central
Hong Kong

Intel Semiconductor of Canada, Ltd.
190 Attwell Drive, Suite 500
Rexdale, Ontario M9W 6H8
Canada


**U.S. INTEL SALES OFFICES** — ***Call (800) 628-8686 to contact any of the following U.S. sales offices:***
**Alabama,** Huntsville; **Arizona,** Phoenix; **California,** Roseville, San Diego, San Jose, Santa Ana, Sherman Oaks; **Colorado,** Denver; **Connecticut,** Danbury; **Florida,** Deerfield Beach, Maitland; **Georgia,** Norcross; **Illinois,** Schaumburg; **Indiana,** Indianapolis; **Maryland,** Annapolis Junction; **Massachusetts,** Westford; **Michigan,** West Bloomfield; **Minnesota,** Bloomington; **New Jersey,** Red Bank; **New York,** Fairport, Fishkill, Islandia; **Ohio,** Beachwood, Dayton; **Oklahoma,** Oklahoma City; **Oregon,** Beaverton; **Pennsylvania,** Blue Bell; **South Carolina,** Columbia, Greenville; **Texas,** Austin, Dallas, Houston; **Utah,** Murray; **Washington,** Bellevue, Spokane; **Wisconsin,** Brookfield

**CANADIAN SALES OFFICES** — ***Call (800) 628-8686 to contact any of the following Canadian sales offices:***
**British Columbia,** Intel Semiconductor of Canada, Ltd., Vancouver
**Ontario,** Intel Semiconductor of Canada, Ltd., Ottawa; Intel Semiconductor of Canada, Ltd., Rexdale
**Quebec,** Intel Semiconductor of Canada, Ltd., Pt. Claire

**U.S. DISTRIBUTORS:**
Alliance Electronics, Inc. / Almac/Arrow Electronics / Arrow Commercial Systems Group / Arrow/Schweber Electronics / Avnet Computer / Hamilton/Avnet / MTI Systems / MTI Systems Sales / North Atlantic Industries Systems Division / Pioneer-Standard / Pioneer Technologies Group / Wyle Laboratories

**CANADIAN DISTRIBUTORS:**
Almac-Arrow Electronics / Arrow Commercial Systems Group / Arrow/Schweber Electronics / Avnet Computer / Hamilton/Avnet / Zentronics

**EUROPEAN INTEL SALES OFFICES:**
**Finland,** Helsinki, (358) 0 544 644
**France,** St. Quentin-en-Yvelines Cedex, (33) (1) 30 57 70 00
**Germany,** Feldkirchen bei Muenchen, (49) 089/90992-0
**Israel,** Tel-Aviv, (972) 03 498080
**Italy,** Assago, (39) (2) 575441
**Netherlands,** Rotterdam, (31) 10 407 11 11
**Russia,** Moscow, 007-095-4439785
**Spain,** Madrid, (34) (1) 308 2552
**Sweden,** Solna, (46) 8 705 5600
**United Kingdom,** Swindon, (44) (0793) 696000

**EUROPEAN DISTRIBUTORS:**
**Austria,** †*Bacher Electronics GmbH, Wien, (43) 1816020
**Belgium,** †*Inelco Distribution, Bruxelles, (32) 2 244 2811; *Diode Belgium, Zaventem, (32) 2 725 46 60
**Denmark,** *Avnet Nortec A/S, Herlev, (45) 4284 2000; †*ITT Multikomponent AS, Glostrup, (45) 4245 6645
**Finland,** †*OY Fintronic AB, Helsinki, (358) 0 682 791
**France,** *Arrow Electronique, Rungis Cedex, (33) (1) 4978 4978; †Metrologie, Asnieres Cedex, (33) (1) 4080 9000; *Tekelec, Sevres, (33) (1) 4623 2425
**Germany,** *Electronic 2000, Muenchen, (49) 89 42110-01; *Jermyn GmbH, Limburg, (49) 6431 5080; †Metrologie GmbH, Muenchen, (49) 89 724470; *Proelectron Vertriebs GmbH, Dreieich, (49) 6103 304343; †Rein Elektronik GmbH, Nettetal, (49) 2153 7330
**Greece,** †Ergodata, Kalithea, (30) 1 95 10 922; *Pouliadis Associates Corp., Athens, (30) 1 36 03 741
**Ireland,** †*Micro Marketing, Dublin, (353) (1) 298 9400
**Israel,** †*Eastronics Limited, Tel-Aviv, (972) 3 6458 777
**Italy,** *Intesi Div. Della Deutsche — Divisione ITT Industries GmbH, Assago (Milano), (39) 2 824701; *Lasi Elettronica, Milano, (39) 2 661431; †Omnilogic Telcom, Milano, (39) 2 48302640
**Netherlands,** †Datelcom, BD Maarsen, (31) 3465 95222; *Diode Components, NG Nieuwegein, (31) 3402 9 12 34; †*Koning en Hartman, AP Delft, (31) 15 609 906
**Norway,** *Avnet Nortec A/S, Hvalstad, (47) 284 6210; †Computer System Integration A/S, Skjetten, (47) 6 84 54 11
**Portugal,** *ATD Electronica LDA, Lisboa, (351) (1) 847 2200; †Metrologia Iberica Portugal, Lisboa, (351) (1) 847 2202

**EUROPEAN DISTRIBUTORS (Contd.):**
**South Africa,** †*EBE, Pretoria, (27) 12 803 7680-93
**Spain,** *ATD Electronica, Madrid, (34) (1) 661 6551; †Metrologia Iberica, Madrid, (34) (1) 661 1142
**Sweden,** †Avnet Computer AB, Farsta, (46) 8 705 18 00; *Avnet Nortec AB, Solna, (46) 8705 1800; *ITT Multikomponent AB, Solna, (46) 8 830020
**Switzerland,** †IMIC Microcomputer, Winkel-Ruti, (41) (1) 8620055; †*Industrade A.G., Wallisellen, (41) (1) 8328111
**Turkey,** *Empa Electronic, Istanbul, (90) (1) 599 3050
**United Kingdom,** *Arrow Electronics, Bedford, (44) 234 270272; *Avnet EMG Ltd., Hertsfordshire, (44) 462 488 500; *Bytech Components, Hants, (44) 256 707 107; †Bytech Systems, Berks, (44) 344 55 333; *Jermyn Electronics, Kent, (44) 732 743 743; †Metrologie VA, Bucks, (44) 494 526 271; *MMD/Rapid Ltd., Berks, (44) 734 750 697

**INTERNATIONAL SALES OFFICES:**
**Australia,** Intel Australia Pty. Ltd., Sydney, 61-2-975-3300; Intel Australia Pty. Ltd., Melbourne, 61-3-810-2141
**Brazil,** Intel Semiconductores do Brazil LTDA, Sao Paulo, 55-11-287-5899
**China/Hong Kong,** Intel PRC Corp., Beijing, (1) 500-4850; Intel Semiconductor Ltd., Hong Kong, (852) 844-4555
**India,** Intel Asia Electronics, Inc., Bangalore, 91-812-215065
**Japan,** Intel Japan K.K., Tsukuba HQ, 0298-47-8511; Intel Japan K.K., Tokyo HQ, 03-3201-3621; Intel Japan K.K., Tokyo, 03-3493-6081; Intel Japan K.K., Kanagawa, 045-474-7660; Intel Japan K.K., Osaka, 06-863-1091; Intel Japan K.K., Hachioji-shi, 0426-48-8770
**Korea,** Intel Korea, Ltd., Seoul, (2) 784-8186
**Mexico,** Intel Tecnologia de Mexico S.A. de C.V., Guadalajara, Jal., 523-640-1259
**Singapore,** Intel Singapore Technology, Ltd., (65) 250-7811
**Taiwan,** Intel Technology Far East Ltd., Taipei, 886-2-5144202

**INTERNATIONAL DISTRIBUTORS:**
**Argentina,** Dafsys S.R.L., Buenos Aires, 54.1334.1871
**Australia,** Email Electronics, Huntingdale, 011-61-3-544-8244; NSD-Australia, Victoria, 03 8900970
**Brazil,** Microlinear, Sao Paulo, 5511-220-2215
**Chile,** Sisteco, Santiago, 562-234-1644
**China/Hong Kong,** Novel Precision Machinery Co., Ltd., Kowloon, Hong Kong, (852) 360-8999
**Guatemala,** Abinitio, Guatemala City, 5022-32-4104
**India,** Priya International Limited, Bangalore, 91-812-214027, 812-214395; Priya International Limited, Bombay, 91-22-2863611, 22-2863676, 22-2863900, 22-2864026; Priya International Limited, New Delhi, 91-11-3314512, 11-3310413; Priya International Limited, Madras, 91-44-451031, 44-451597; Priya International Limited, Secunderabad, 91-842-813549, 842-813120; SES Computers and Technologies Pvt. Ltd., Delhi, 91-11-6849285, 11-6843006; SES Computers and Technologies Pvt. Ltd., Bombay, 91-22-4939977, 22-4943731; SES Computers and Technologies Pvt. Ltd., Bangalore, 91-812-348481, 812-343685
**Jamaica,** MC Systems, Kingston, (809) 926-0104
**Japan,** Asahi Electronics Co. Ltd., Kitakyushu-shi, 093-511-6471; CTC Components Systems Co., Ltd., Kanagawa 213, 044-852-5121; Dia Semicon Systems, Inc., Tokyo 154, 03-3439-1600; Okaya Koki, Nagoya-shi, 052-204-8315; Ryoyo Electro Corp., Tokyo 104, 03-3546-5011
**Korea,** J-Tek Corp., Seoul, (822) 557-8039; Samsung Electronics, Seoul, (822) 751-3680
**Mexico,** PSI S.A. de C.V., Cuernavaca-MOR, 52-73-11-1994/5
**New Zealand,** Email Electronics, Penrose, Auckland, 011-64-9-591-155
**Saudi Arabia,** AAE Systems, Inc., Sunnyvale, CA U.S.A., (408) 732-1710
**Singapore,** Electronic Resources Pte, Ltd., (65) 283-0888
**South Africa,** Electronic Building Elements, Pretoria, 011-2712-803-7680
**Taiwan,** Micro Electronics Corp., Taipei, R.O.C., (886) 2-7198419; Acer Sertek, Inc., Taipei, R.O.C., (886) 2-501-0055
**Uruguay,** Interfase, Montevideo, 5982-49-4600
**Venezuela,** Unixel C.A., Caracas, 582-238-7749

*Technical Distributor
†VAD